patchDPCC: A Patchwise Deep Compression Framework for Dynamic Point Clouds

Zirui Pan[1], Mengbai Xiao[1*], Xu Han[1], Dongxiao Yu[1], Guanghui Zhang[1], Yao Liu[2]
[1]Shandong University
[2]Rutgers University
panzirui@mail.sdu.edu.cn, xiaomb@sdu.edu.cn, hanx@mail.sdu.edu.cn,
dxyu@sdu.edu.cn, gh.zhang@sdu.edu.cn, yao.liu@rutgers.edu

Abstract

When compressing point clouds, point-based deep learning models operate points in a continuous space, which has a chance to minimize the geometric fidelity loss introduced by voxelization in preprocessing. But these methods could hardly scale to inputs with arbitrary points. Furthermore, the point cloud frames are individually compressed, failing the conventional wisdom of leveraging inter-frame similarity. In this work, we propose a patchwise compression framework called patchDPCC, which consists of a patch group generation module and a point-based compression model. Algorithms are developed to generate patches from different frames representing the same object, and more importantly, these patches are regulated to have the same number of points. We also incorporate a feature transfer module in the compression model, which refines the feature quality by exploiting the inter-frame similarity. Our model generates point-wise features for entropy coding, which guarantees the reconstruction speed. The evaluation on the MPEG 8i dataset shows that our method improves the compression ratio by 47.01% and 85.22% when compared to PCGCv2 and V-PCC with the same reconstruction quality, which is 9% and 16% better than that D-DPCC does. Our method also achieves the fastest decoding speed among the learning-based compression models.

Introduction

Dynamic point clouds are essential to emerging virtual reality (VR) applications like immersive filmmaking[1] and interactive advertising.[2] To offer a truly engaging and captivating experience, millions of points must be rendered, which challenges storing, transmitting, and processing the dynamic point cloud. Therefore, effectively compressing dynamic point clouds is crucial to deploying these VR applications in the real world.

A dynamic point cloud could be compressed with deep learning (DL) in an end-to-end manner. Compared to the rule-based methods like Moving Picture Experts Group (MPEG) standards (Schwarz et al. 2019), the DL-based methods usually achieve higher compression ratios. The DL-based methods are categorized into voxel-based ones (Quach, Valenzise, and Dufaux 2020; Wang et al. 2021b,a), octree-based ones (Huang et al. 2020; Que, Lu, and Xu 2021; Fu et al. 2022), and point-based ones (Huang and Liu 2019; Gao et al. 2021). In the voxel-based methods, the 3D space is split into blocks that are compressed individually. A block is further voxelized, and the compression task is turned into a classification task checking if voxels are occupied. The octree-based methods convert a point cloud into an octree (Laboratory and Meagher 1980) first and focus on optimizing the following entropy coding with a learning-based model (Mekuria, Blom, and César 2017; Kammerl et al. 2012). However, such methods have to align the point coordinates to the voxels or the tree nodes in preprocessing, leading to geometric information losses. On the other hand, the point-based methods directly accept point coordinates and extract latent features to compress. By processing points in the continuous space, the geometric fidelity losses caused by voxelization could be avoided. However, the point-based methods can hardly scale to arbitrary points since a trained model must process fixed-size inputs. Furthermore, most learning-based methods are designed to compress static point clouds, failing to exploit the high similarity between consecutive frames in a dynamic point cloud.

In this work, we propose patchDPCC, a deep compression framework for dynamic point clouds. Our framework consists of a patch group generation module followed by a deep compression model, which is point-based so that the reconstruction quality is guaranteed. The patch group generation module is designed to separate point cloud frames into fixed-size patches, enabling our framework to compress arbitrary points. However, it is challenging to divide consecutive point cloud frames into patches while also exploiting the inter-frame similarity. If we divide point clouds individually, the patches from different frames can hardly represent the same part of an object. To solve this, we organize point cloud frames into groups of frames (GoFs), where there are a leading I-frame and the following P-frames. The I-frame is splitted into fixed-size patches individually. By referencing the I-frame patches, algorithms are developed to divide the P-frames into patches with high similarity. More importantly, all patches have the same number of points, and the ones representing the same part of an object in different frames are grouped together. Our deep compression model accepts a patch group to compress, where a feature transfer module

*Corresponding author.

[1]https://www.youtube.com/watch?v=iwUkbi4_wWo
[2]https://www.anayi.com/include_html/4d/21ss_4d_11.html

is installed to extract patch features by leveraging the features from the previous patch. The feature transfer module both impoves the reconstruction quality and decoding time. Additionally, since the size of patch groups is controlled, extracting point-wise features becomes bearable, which further speeds up the decoding process as the time-consuming down-/up-sampling operations are eliminated.

In the evaluation, we compress dynamic point clouds in the 8i dataset (d'Eon et al. 2017) published by MPEG, which contains nearly one million points in a point cloud frame. Our method outperforms the state-of-the-art methods, i.e., G-PCC, V-PCC, PCGCv2 (Wang et al. 2021a) (voxel-based), OctAttention (Fu et al. 2022) (octree-based), and D-DPCC (Fan et al. 2022) (voxel-based), in rate-distortion tradeoffs and BD-rate gains. In terms of the decoding time, patchDPCC is the fastest among learning-based methods.

The contributions of our work are as follows:

- We design a patch group generation module that generates fixed-size and temporally correlated patch groups. This helps the point-based compression model process point clouds at arbitrary sizes.
- We propose a point-based compression module that leverages inter-frame correlation and point-wise features to improve reconstruction quality and decoding speed.
- Our solutions form a practical compression framework for dynamic point clouds, which outperforms other state-of-the-art compression methods in the evaluation.

Related Work

Learning-based Point Cloud Compression

Deep learning techniques have been experimented to compress point clouds and show their efficacy. The self-supervised autoencoder structure is noticed well suited for point cloud compression (Kramer 1991, 1992), where the latent features are compact, and the decoder could reconstruct the point cloud. Quach et. al. (Quach, Valenzise, and Dufaux 2019) for the first time introduce a well-designed deep learning model for point cloud compression. The following studies (Quach, Valenzise, and Dufaux 2020; Wang et al. 2021b,a; Huang et al. 2020; Que, Lu, and Xu 2021; Fu et al. 2022; Huang and Liu 2019; Gao et al. 2021) show that the learning-based compression methods reach higher compression ratios than the hand-crafted ones. Some of the studies (Quach, Valenzise, and Dufaux 2020; Wang et al. 2021b) split the point cloud into small enough blocks and voxelize these blocks. The voxelized blocks are learned with 3D convolutions, and the decoding process becomes predicting the occupancy of voxels. The voxel-based methods achieve high compression ratios but are also time- and memory-consuming. Moreover, the models are sensitive to density changes and may fail the compression of sparse point clouds. The voxelization also introduces precision losses in reconstruction. With the development of 3D sparse convolution (Choy, Gwak, and Savarese 2019), Wang et. al. (Wang et al. 2021a) propose an end-to-end network accepting an input point cloud at a large scale, which achieves the state-of-the-art performance in compressing static point clouds. Another thread of studies (Huang et al. 2020; Que, Lu, and Xu 2021; Fu et al. 2022) encode a point cloud into an octree first and employ learning-based entropy coding to optimize the compression. However, the compression ratio is bound to the octree, which can hardly be as high as the voxel-based ones. There are also studies (Huang and Liu 2019; Gao et al. 2021) that directly compress and reconstruct the point cloud coordinates. Nevertheless, it is not easy to deploy the point-based methods because the point number varies significantly in practical point clouds.

Dynamic Point Cloud Compression

A number of rule-based compression methods have been developed towards dynamic point clouds. The MPEG standard V-PCC (Schwarz et al. 2019) projects points from the 3D space onto 2D images, and further compress them with traditional video codecs (Li et al. 2020). Although V-PCC achieves high compression ratios, the complex pipeline results in long compression and decompression time. There are also approaches (Chen et al. 2023; Mekuria and César 2016; Mekuria, Blom, and Cesar 2017) designed to directly remove inter-frame redundancy in the 3D space. They detect motions by comparing blocks or patches between neighboring frames. Most learning-based compression methods could be directly applied to compress a dynamic point cloud: Every point cloud frame in the sequence is compressed individually. But this fails the conventional wisdom of video coding that inter-frame similarity should be well leveraged. D-DPCC (Fan et al. 2022) is the only learning-based compression solution for dynamic point clouds, which exploits motion estimation and compensation in the feature space.

Methodology

Problem Definition A dynamic point cloud is a sequence of point sets $\{\mathcal{P}_i\}$. $\mathcal{P}_i$ is the i-th frame with X_i points. To compress a dynamic point cloud, point features F_i are extracted followed by quantization and entropy coding, generating a compact bitstream. At the decoder side, the bitstream is recovered to quantized features $\hat{F}_i$, which are then reconstructed to point cloud frames $\hat{\mathcal{P}}_i$ again.

Overall Architecture Inspired by conventional video codecs like H.264 (Wiegand et al. 2003), we divide a point cloud sequence into groups of frames (GoFs), and each GoF has a leading I-frame and subsequent P-frames. In our compression framework, a GoF is first separated into *patch groups*, and the patch groups are compressed by *a compression model*. The compression model is also responsible for recovering the patch groups from the compression domain, after which the GoF is trivially synthesized. Figure 1 shows the overall architecture of our compression framework.

Patch Group Generation

To exploit the inter-frame similarity, we need not only generate patches from individual frames but also group patches representing the same part of an object from different frames in a GoF. More importantly, these patches should contain the same number of points to align with the following compression model. We generate patch groups among frames of

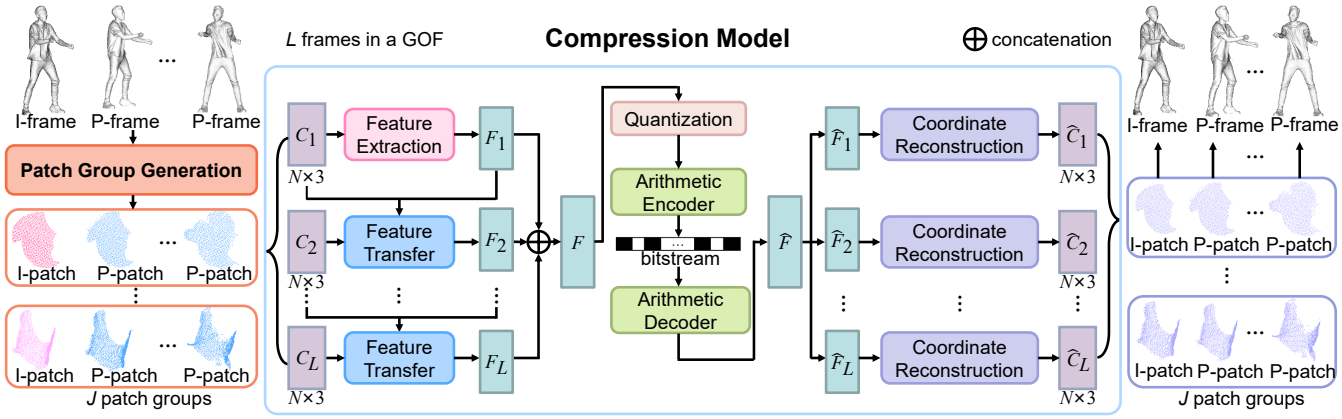

Figure 1: The overview of patchDPCC.

a GoF in three stages: *generating I-patches*, *generating P-patches*, and *adjusting P-patches*.

Generating I-Patches We first generate patches having exact N points from the I-frame, where N is the input dimension our compression model expects to accept. These patches are named *I-patches*. For an I-frame containing X points, we randomly delete $(X \mod N)$ points. Then, we follow the method of generating fixed-size patches described in a prior study (Guarda, Rodrigues, and Pereira 2021): 1) the I-frame is divided into patches with roughly the same size according to local densities, then 2) the points are moved from patches having points exceeding N to their neighboring patches with points less than N, which is an iterative process. 3) While all patches are at the size of N, points are exchanged between neighboring patches to avoid overlapping. To the end, we would generate $J = \lfloor X/N \rfloor$ I-patches, each having N points.

Generating P-Patches To generate patches in a P-frame, i.e., *P-patches*, we apply the iterative closest point (ICP) algorithm from the I-patches to it. As a result, the I-patches are transformed to fit the P-frame. For each point in the P-frame, we associate it to the j-th transformed I-patch that has its nearest neighbor as

$$\operatorname*{argmin}_j \mathrm{d}(p, M_j \mathbf{P}_j^I), j = 1, \ldots, J,$$

where p is the point, $\mathbf{P}_j^I$ is a point set representing the j-th I-patch, and M_j is the transformation matrix calculated by the ICP algorithm. $d(\cdot)$ calculates the shortest distance between a given point and a point set $\mathbf{P}$ as

$$\mathrm{d}(p, \mathbf{P}) = \min_{p' \in \mathbf{P}} \|p - p'\|,$$

As a result, the P-frame is also divided into J patches, and each is paired with one of the I-patches. We generate P-patches for all P-frames. An I-patch and its associated P-patches form a patch group, and now we have J patch groups. However, the point number in a P-patch might not equal N. In order to make the P-patches also have the exact N points, we need either add points to or delete points from the P-patches.

Adjusting P-Patches For a P-patch $\mathbf{P}^P$ having points less than N, we directly copy points from the transformed I-patch $\mathbf{P}^{I_t}$ in the same patch group. Since we adjust the P-patch only for aligning with the following compression model, its geometric structure should not be substantially changed. We follow three principles when copying points: 1) We always select a transformed I-patch point from where there are also P-patch points, 2) a point with high local density is preferred, and 3) the points copied should be evenly distributed. To achieve this, we fuse the P-patch and the transformed I-patch, and build an octree over them. On any branch, the tree building stops either there is no point in the subspace or the leaf node reaches a resolution of r. When copying a point, we choose the tree node having the most I-patch points and any P-patch point, and the candidate point is the one that has the nearest P-patch neighbor. As long as the point is copied, the point selection will not be carried out in this node and its neighboring nodes within $2r$ in the near future. We repeatedly select and copy points until the P-patch has N points. The left side of Figure 2 gives an example of copying points to the P-patch.

Occasionally, there are not enough points to copy in the tree nodes having points from both patches. In this case, we directly copy all I-patch points in these nodes and select the remaining points from those with only I-patch points. We check the points in these nodes and copy the one having the nearest P-patch neighbors until the P-patch has exact N points. The complete algorithm of copying points to the P-patch is shown in Appendix[3].

For a P-patch with points greater than N, we always select a leaf node with the most P-patch points and delete one random point from it. Also, we will not delete a point from this node and its neighboring nodes within $2r$ in the near future. The deletion is repeated until the P-patch has exact N points. We present an example of deleting points in the P-patch at the right side of Figure 2 and the algoirthm in Appendix. To the end, for a GoF, we generate J patch groups, where the patches are composed of N points. It is worth noting that the transformed I-patch is only used to split and adjust

[3]Our Appendix is at https://github.com/pzrsdu/patchDPCC

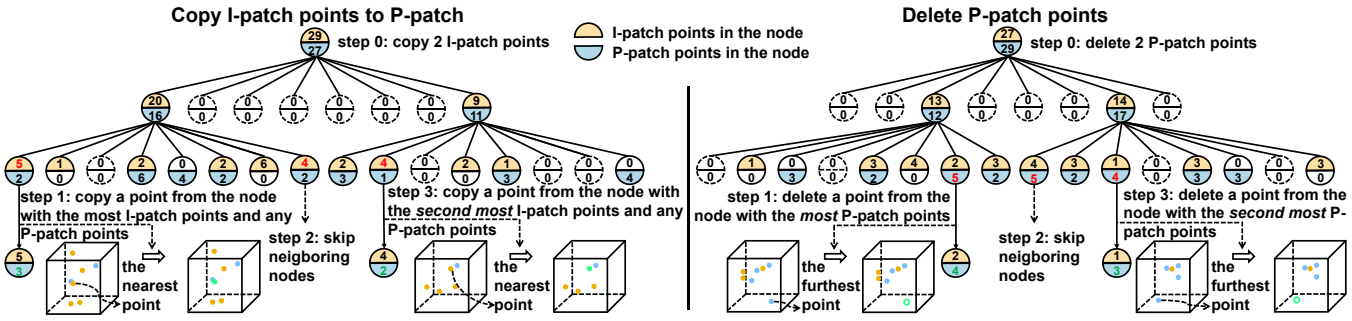

Figure 2: Two examples of adjusting P-patches. The octrees are built upon points fused from a P-patch and the corresponding transformed I-patch. To align with the point number of I-patch, we either copy 2 points (left) to or delete 2 points from the P-patch.

the P-patches, and the original I-patch is encapsulated in the patch group for following compression.

Compression Model

Our compression model is based on the autoencoder structure, while the encoder accepts a patch group which is represented by its coordinates C in $L \times N \times 3$. In the patch group, the I-patch C_1 is digested by a *feature extraction* module, and the P-patches $C_i, i = 2, \ldots, L$ are sent to a *feature transfer* module. Both modules output point-wise features and they are concatenated to F in $L \times N \times G$. F is quantized to $\hat{F}$, which is further compressed by entropy coding. At the decoder side, we decompress the quantized features and recover the point cloud patches $\hat{C}$ in a *coordinate reconstruction* module.

Feature Extraction The feature extraction module takes an I-patch as the input, and outputs its point-wise feature F_0 in $N \times G$. We adopt a network structure (Wang et al. 2019) that extracts features with four densely connected blocks. In each block, data are processed via feature-based k-nearest neighbors (k-NN), a chain of densely connected multilayer perceptrons (MLPs), and max pooling. This structure is proved effective and efficient compared to downsampling-based networks (Li, Chen, and Lee 2018; Qi et al. 2017), in which costly search for point correspondences between layers is required.

Feature Transfer In a patch group, patches from adjacent frames are highly similar, so we propose to *transfer* features of a patch to the subsequent one. To generate the feature F_i of a P-patch in $N \times G$ where $i = 2, \ldots, L$, the inputs of the feature transfer module are the P-patch coordinates C_i, the coordinates of the previous I-/P-patch in C_{i-1}, and the previous patch feature F_{i-1}.

The design of our feature transfer module is presented in Figure 3. We consider F_{i-1} from the previous frame as a template that could be fine-tuned to obtain F_i. To achieve this, F_{i-1} is concatenated with C_i, and they are fed to an MLP of three layers, extracting the initial feature $\tilde{F}_i$. On the other hand, for each point in the current patch C_i, we conduct k-NN search in C_{i-1} and group them to capture more local information from the previous patch, which expands

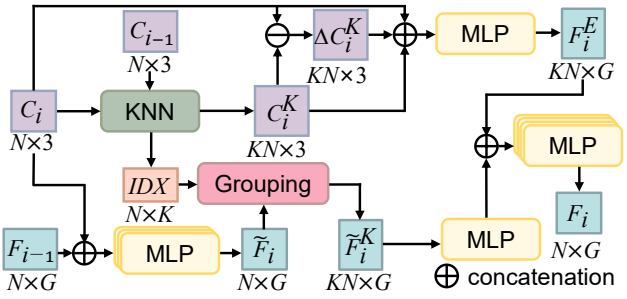

Figure 3: The network structure of feature transfer module

the coordinates to C_i^K in $KN \times 3$ and gets the neighboring indices IDX in $N \times K$. With IDX, we duplicate and group features in $\tilde{F}_i$ to $KN \times G$ as $\tilde{F}_i^K$.

We further refine $\tilde{F}_i^K$ with the absolute and relative coordinates in local fields. For each point in C_i, its nearest neighboring points in the previous patch have been grouped in C_i^K, so the coordinate residuals of neighbors from this point could be calculated in ΔC_i^K. By concatenating ΔC_i^K, C_i, and C_i^K and sending them to an MLP, we generate features F_i^E in $KN \times G$ that encode positional information of the current patch. To the end, we concatenate F_i^E to $\tilde{F}_i^K$ and generate the final P-patch feature F_i. Refining features of a P-patch with positional coding could be expressed as

$$F_i = MLP_3(MLP_1(C_i \oplus \Delta C_i^K \oplus C_i^K) \oplus MLP_2(\tilde{F}_i^K)),$$

where MLP_1 and MLP_2 are single-layered, MLP_3 are four-layered, and $\oplus$ is the concatenation operation. As the feature quality of P-patches have been improved by the feature transfer, we need not distinguish them from the I-patch in reconstruction.

Entropy Coding The features of patches in a patch group could be further compressed. The extracted features are concatenated to F, which is then quantized to $\hat{F}$. The quantization is simply rounding floating-point numbers in the features to the nearest integers. After quantization, we apply an arithmetic coder to encode $\hat{F}$ with the probability distribution estimated by a factorized entropy model (Ballé, Laparra,

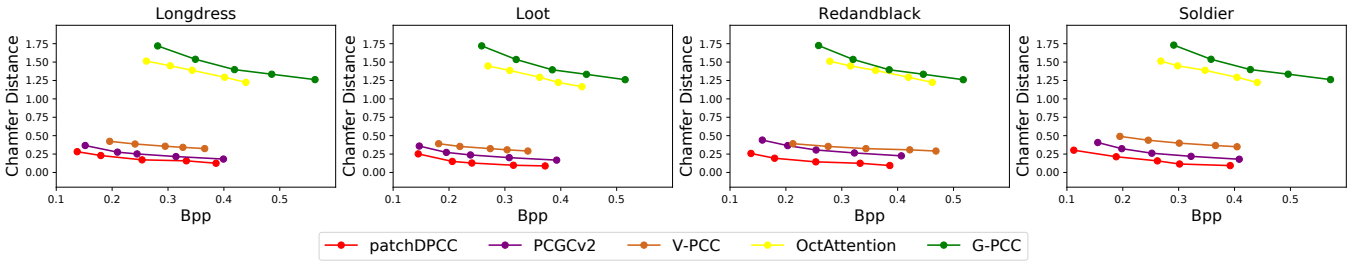

Figure 4: Chamfer Rate-distortion curves of different compression methods on MPEG 8i

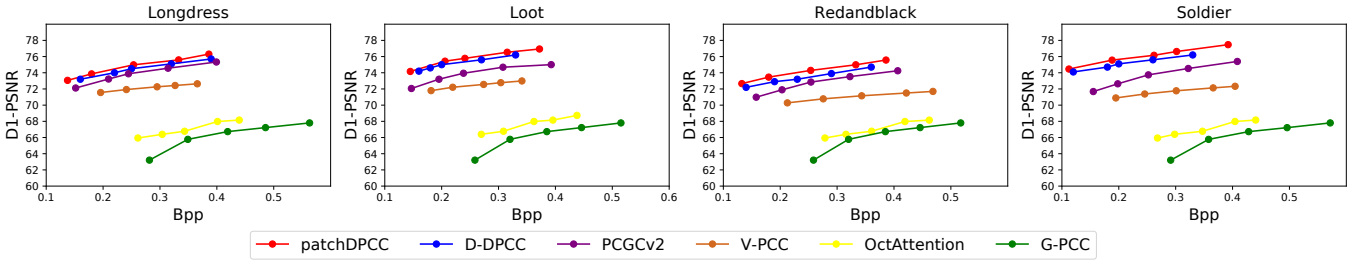

Figure 5: D1-PSNR Rate-distortion curves of different compression methods on MPEG 8i

and Simoncelli 2017), where one entropy bottleneck layer is used and all patch groups share the same entropy model. At the decoder side, the encoded bitstream is decompressed into $\hat{F}$ again, which is prepared for patch group reconstruction.

Coordinate Reconstruction With the decompressed features, MLPs could be used to reconstruct the point coordinates. However, the reconstructed points of P-patches often deviate from their proper positions because of the feature transfer. So we adopt the self-attention mechanism (Vaswani et al. 2017) to explore inter-feature dependencies and reserve the most suitable ones. Specifically, we first individually reconstruct the coarse coordinates of patches $\hat{C}_i'$ from $\hat{F}_i$ via a three-layered MLP. For each patch, we further concatenate $\hat{C}_i'$ and $\hat{F}_i$ and feed them into a self-attention unit (Zhang et al. 2019), extracting the refined point-wise features $\hat{F}_i'$ in $N \times G$. $\hat{F}_i'$ is expected to regress attention weights of all points for long-range context dependencies. We reconstruct the final patch $\hat{C}_i$ via a three-layered MLP from $\hat{F}_i'$.

Loss Function We employ the rate-distortion joint loss function for end-to-end optimization as follows

$$L = R + \lambda D,$$

where R is the *rate* measured in bits per point (bpp), and D is the *distortion* that is jointly defined by multiple metrics measuring the quality fidelity between the input point cloud and the reconstructed one.

As the rounding operation before entropy coding is not differentiable, we replace it with adding a uniform noise $\mathcal{N} \sim (-0.5, 0.5)$ during the training, which simulates the loss caused by quantization. The compression rate R in bpp is derived from the probabilities of features estimated by an entropy model (Ballé, Laparra, and Simoncelli 2017).

In a compression task, the decompressed point cloud should be identical to the input one. Thus, the distortion D is a metric that reflects the quality fidelity between the input and the output. We use two point-to-point metrics, Chamfer Distance (CD) (Fan, Su, and Guibas 2017) and Hausdorff Distance (HD) (Berger et al. 2013), in the distortion definition. Additionally, we also incorporate quality fidelity between the input point cloud and the coarse reconstruction result $\hat{C}_i'$, which would also benefit the final reconstruction quality. The distortion D is thus defined as

$$D = \mathrm{CD}(C_i, \hat{C}_i') + \mathrm{HD}(C_i, \hat{C}_i') + \alpha(\mathrm{CD}(C_i, \hat{C}_i) + \mathrm{HD}(C_i, \hat{C}_i)),$$

where $\mathrm{CD}(\cdot)$ and $\mathrm{HD}(\cdot)$ evaluate CD and HD between two point clouds, respectively. We gradually scale up the coefficient α during training so that the model pays more attention to reconstruction details in later epochs.

Experiment

Experimental Setup

Baselines. We compare patchDPCC with the peer compression methods. For the rule-based ones, we set up the MPEG standards G-PCC test model v14.0 and V-PCC test model v18.0 (Schwarz et al. 2019). For the learning-based ones, patchDPCC is compared with PCGCv2 (Wang et al. 2021a), OctAttention (Fu et al. 2022), and D-DPCC (Fan et al. 2022). For methods designed to compress static point clouds, i.e., G-PCC, PCGCv2, OctAttention, they are used to compress the point cloud frames in the experiments individually. D-DPCC is a model also designed to compress dynamic point clouds. While the source code of D-DPCC is not publicly available, we keep our experimental setup consistent with theirs and compare our results with the numbers reported in the paper (Fan et al. 2022).

Training Dataset. We train patchDPCC, PCGCv2, OctAttention with the *Owlii* dataset (Xu, Lu, and Wen 2017),

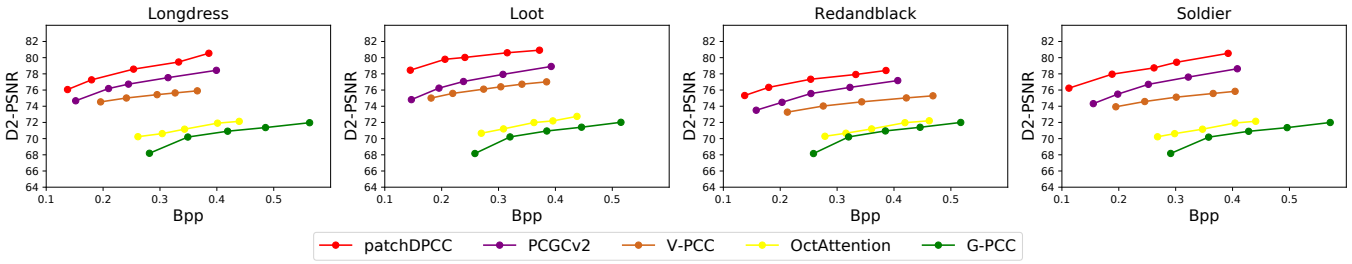

Figure 6: D2-PSNR Rate-distortion curves of different compression methods on MPEG 8i

	patchDPCC				D-DPCC			
	PCGCv2		V-PCC		PCGCv2		V-PCC	
	D1-PSNR	D2-PSNR	D1-PSNR	D2-PSNR	D1-PSNR	D2-PSNR	D1-PSNR	D2-PSNR
Longdress	-37.65%	-33.19%	-81.11%	-76.12%	-22.31%	-17.91%	-78.92%	-72.89%
Loot	-53.35%	-50.69%	-85.03%	-86.83%	-36.15%	-29.73%	-71.76%	-70.92%
Redandblack	-39.32%	-39.32%	-84.96%	-79.08%	-25.68%	-20.56%	-68.97%	-76.71%
Soldier	-57.72%	-45.59%	-89.76%	-91.31%	-39.48%	-36.47%	-85.71%	-73.08%
Average	**-47.01%**	**-42.21%**	**-85.22%**	**-83.33%**	**-31.14%**	**-26.39%**	**-76.66%**	**-74.43%**

Table 1: RD-Rate gains of patchDPCC (D-DPCC) against PCGCv2 and V-PCC

which contains 4 point cloud sequences. Each sequence is at 30 frames per second and lasts 20 seconds, thus comprising 600 frames in total. Following the setting of D-DPCC, we quantize the dataset that decreases the precision from 11-bit to 9-bit for saving time and memory.

Testing Dataset. We test the learning-based and hand-crafted rule-based models both with the *MPEG 8i* dataset (d'Eon et al. 2017), which includes 4 point cloud sequences with the precision of 10-bit. Each point cloud sequence renders a moving person at 30 frames per second and lasts 10 seconds.

Training Strategy. To obtain different tradeoffs between the compression ratio and the reconstruction quality, we set λ in the loss function as one of $\{1,\ 1.5,\ 2,\ 2.5,\ 3\}$ and train multiple compression models. The training parameter α linearly increases from 0.001 to 1, so the reconstruction focuses more on details at later epochs. In the training stage, the batch size is 5, and we use Adam optimizer (Kingma and Ba 2015) with a learning rate decaying from 10^{-3} to 10^{-6}. We set K to 16 in neighbor searching, N to 2048 as the patch size, and G to 256 as the feature dimension. The training and testing of all methods are carried out on a single server equipped with an NVIDIA RTX3090 GPU with 24GB GDDR memory. GoF size L is 5.

Evaluation Metrics. We measure bits per point (bpp) representing the compression ratio, which is calculated by dividing the bitstream size after entropy coding by the input point number. As for the distortion, we follow MPEG common test condition (CTC) that defines point to point PSNR (D1-PSNR) and point to plane PSNR (D2-PSNR), which quantifies how much a point cloud is modified from a reference. We also measure Chamfer Distance (CD) to evaluate the reconstruction quality. For D1-PSNR and D2-PSNR, we set PSNR peak value to 1023 following D-DPCC.

Rate vs. Distortion

To plot RD-curves of various compression methods, we have to collect multiple performance points. For PCGCv2, we adjust a parameter in their loss function that controls the rate-distortion tradeoffs. The parameter is similar to λ of our model. We set it as 2, 3, 4, 7, 10, and train compression models accordingly. For V-PCC, we set the quantization parameter (QP) as 11, 12, 13, 15, 18. In G-PCC, the position quantization scale parameter is set to 0.5, 0.55, 0.6, 0.65, 0.7 to reflect the tradeoffs between rate and distortion. We set the quantization step parameter as 1.4, 1.5, 1.6, 1.7, 1.8 for OctAttention.

For distortion measured by CD, D1-PSNR, D2-PSNR, we plot the RD-curves of various compression methods in Figure 4, 5 and 6 respectively. To give a sense of how patchDPCC is compared with D-DPCC, we also plot the data points of D-DPCC in Figure 5, which are roughly estimated by observing figures in the original paper (Fan et al. 2022) (the RD-curves of CD and D2-PSNR are not presented in the original paper of D-DPCC). We notice that patchDPCC outperforms all other schemes in both the compression ratio and the reconstruction quality. While OctAttention and G-PCC are octree-based methods, their compression ratios are bound to the octrees built over the point clouds. Only by sacrificing reconstruction quality to a quite low level, their compression ratio in bpp could approximate the other methods. As a hand-crafted rule-based method, V-PCC achieves lower bitrates and higher reconstruction quality than the octree-based ones since it effectively takes advantage of the inter-frame redundancy in point cloud sequences with conventional video codecs. PCGCv2 gains superior performance than V-PCC with the assistance of a deep learning structure. For D-DPCC, another learning-based compression model

	CD		D1-PSNR (dB)		D2-PSNR (dB)		Enc. Time (s)		Dec. Time (s)	
	w/ FT.	w/o FT.	w/ FT.	w/o FT.	w/ FT.	w/o FT.	w/ FT.	w/o FT.	w/ FT.	w/o FT.
Longdress	0.12	0.18	76.31	74.58	80.54	76.74	4.49	4.56	1.22	1.21
Loot	0.08	0.17	76.93	74.26	80.92	78.51	4.74	4.82	1.24	1.24
Redandblack	0.09	0.14	75.57	73.96	78.41	74.24	4.35	4.48	1.23	1.22
Soldier	0.09	0.11	77.47	75.62	80.53	77.84	6.66	6.82	1.70	1.69
Average	**0.11**	**0.15**	**76.56**	**74.61**	**80.10**	**76.83**	**5.06**	**5.17**	**1.34**	**1.34**

Table 2: Ablation study for feature transfer module in patchDPCC

	Average Bpp			Time (s)	
N	$\lambda = 1$	$\lambda = 2$	$\lambda = 3$	Enc. Time	Dec. Time
1024	1.04	1.59	2.85	5.12	1.41
2048	**0.13**	**0.25**	**0.38**	5.06	1.34
4096	0.52	1.27	2.13	5.02	1.29
8192	1.24	1.58	2.51	4.97	1.25
10240	1.84	2.72	3.23	**4.88**	**1.19**

Table 3: Impacts of Patch Size N in patchDPCC

that also exploits inter-frame similarity, it is slightly worse than our method when measuring D1-PSNR of four test videos.

We also calculate the quantitative results in BD-rate gains by comparing our method with PCGCv2 and V-PCC. We do not report the BD-rate gains against G-PCC and OctAttention because the performance gap between them and patchDPCC is too large that the BD-rate gains cannot be effectively calculated. The results are shown in Table 1, where we also list the results reported by D-DPCC aside for comparison. We can see that when compared with V-PCC, patchDPCC reduces the average compression sizes by 85.2% and 83.3% with the same D1-PSNR and D2-PSNR, respectively. For the same dataset, D-DPCC reduces the compression sizes by 76.66% and 74.43% against V-PCC. When comparing with PCGCv2, patchDPCC improves the BD-Rate gains by 47% (PSNR-D1) and 42.2% (PSNR-D2), while D-DPCC improves the compression ratios by 31.14% (PSNR-D1) and 26.39% (PSNR-D2). Comparing the BD-rate gains reported by D-DPCC against PCGCv2 and V-PCC, we can find that patchDPCC is 9% and 16% better.

Ablation Study

Effectiveness of Feature Transfer. To evaluate the effectiveness of the feature transfer module, we replace all feature transfer modules with feature extraction modules. The results are presented in Table 2. Without the feature transfer module, the average CD is increased from 0.1004 to 0.1538 (34.72% higher). The other two quality metrics, average D1-PSNR and average D2-PSNR, are lowered by 2.6% and 4.25%, respectively. This verifies that our feature transfer module helps improve the feature quality by using the feature from the previous frame. We also report the time of encoding and decoding a frame by our compression model. We accumulate the time used to encoding and decoding patch groups of a GoF individually, and average them by the GoF size L. Replacing the feature transfer module with the feature extraction module increases the encoding time by 2.1%, which shows that the feature transfer operation is more efficient than the normal feature extraction.

Various Patch Sizes

We also evaluate how different patch sizes impact the performance of our model. In the experiments, we alter the patch size N and train our compression model accordingly. The patch size is selected from one of 1024, 2048, 4096, 8192 and 10240 points. For each patch size, we also vary λ in the loss function to observe different rate-distortion tradeoffs. The experimental results are shown in Table 3. We can see that when N is 2048, patchDPCC reaches the lowest average bpp, which are 0.13, 0.25 and 0.38, respectively. Though increasing the patch to 10240 helps save the encoding time by 3.6% and the decoding time by 11.2%, it also leads to 8 times higher bpp. As a result, we always set the patch size N as 2048 in other experiments.

Coding Time

patchDPCC implements a single-threading version of the algorithm in the prior study (Guarda, Rodrigues, and Pereira 2021), and it takes about 10 minutes to split an I-frame into I-patches. Generating P-patches takes around 1.5 seconds for each P-frame. While focusing on the encoding time of compression models only, patchDPCC takes 5.06s per frame. Other methods use 64.79s (V-PCC), 1.27s (G-PCC), 1.27s (PCGCv2) and 0.74s (OctAttention). For decoding a frame from the compression domain, patchDPCC needs 1.34s, while other methods spend 2.41s (V-PCC), 0.87s (G-PCC), 5.91s (PCGCv2) and 793.06s (OctAttention). patchDPCC is only slower than G-PCC, and achieves the fastest decoding time among learning-based methods.

Conclusion

In this paper, we propose patchDPCC, a novel deep compression framework comprising a patch group generation module and a deep compression model. The patch group generation module splits point cloud frames into fixed-size patches and groups the ones representing the same object. This enables the compression model to exploit inter-frame similarity and process frames with arbitrary points. Evaluation shows that patchDPCC outperforms state-of-the-art compression methods in BD-rate gains and decoding speed.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 62102229), Natural Science Foundation of Shandong Province, China (No. ZR2022ZD02), Shandong Excellent Young Scientists Fund Program (Overseas) (No. 2023HWYQ-045), Natural Science Foundation of Qingdao (No. 23-2-1-127-zyyd-jch) and National Natural Science Foundation of China (No. 62302268).

References

Ballé, J.; Laparra, V.; and Simoncelli, E. P. 2017. End-to-end Optimized Image Compression. In *Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017*.

Berger, M.; Levine, J. A.; Nonato, L. G.; Taubin, G.; and Silva, C. T. 2013. A benchmark for surface reconstruction. *ACM Trans. Graph.*, 32(2): 20:1–20:17.

Chen, R.; Xiao, M.; Yu, D.; Zhang, G.; and Liu, Y. 2023. patchVVC: A Real-time Compression Framework for Streaming Volumetric Videos. In *Proceedings of the 14th Conference on ACM Multimedia Systems, MMSys 2023, Vancouver, BC, Canada, June 7-10, 2023*, 119–129.

Choy, C. B.; Gwak, J.; and Savarese, S. 2019. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *Proceedings of the 32nd IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 3075–3084.

d'Eon, E.; Harrison, B.; Myers, T.; and Chou, P. A. 2017. 8i Voxelized Full Bodies - A Voxelized Point Cloud Dataset. ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006.

Fan, H.; Su, H.; and Guibas, L. J. 2017. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2463–2471.

Fan, T.; Gao, L.; Xu, Y.; Li, Z.; and Wang, D. 2022. D-DPCC: Deep Dynamic Point Cloud Compression via 3D Motion Prediction. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, July 23-29, 2022*, 898–904.

Fu, C.; Li, G.; Song, R.; Gao, W.; and Liu, S. 2022. OctAttention: Octree-Based Large-Scale Contexts Model for Point Cloud Compression. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence, AAAI 2022, 34th Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, 12nd Symposium on Educational Advances in Artificial Intelligence, EAAI 2022, Virtual Event, February 22-March 1, 2022*, 625–633.

Gao, L.; Fan, T.; Wan, J.; Xu, Y.; Sun, J.; and Ma, Z. 2021. Point Cloud Geometry Compression Via Neural Graph Sampling. In *Proceedings of the 28th IEEE International Conference on Image Processing, ICIP 2021, Anchorage, AK, USA, September 19-22, 2021*, 3373–3377.

Guarda, A. F. R.; Rodrigues, N. M. M.; and Pereira, F. 2021. Constant Size Point Cloud Clustering: A Compact, Non-Overlapping Solution. *IEEE Trans. Multim.*, 23: 77–91.

Huang, L.; Wang, S.; Wong, K.; Liu, J.; and Urtasun, R. 2020. OctSqueeze: Octree-Structured Entropy Model for LiDAR Compression. In *Proceedings of the 33rd IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 1310–1320.

Huang, T.; and Liu, Y. 2019. 3D Point Cloud Geometry Compression on Deep Learning. In *Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, October 21-25, 2019*, 890–898.

Kammerl, J.; Blodow, N.; Rusu, R. B.; Gedikli, S.; Beetz, M.; and Steinbach, E. G. 2012. Real-time compression of point cloud streams. In *Proceedings of the 29th IEEE International Conference on Robotics and Automation, ICRA 2012, St. Paul, Minnesota, USA, May 14-18, 2012*, 778–785.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015*.

Kramer, M. 1992. Autoassociative neural networks. *Computers & Chemical Engineering*, 16(4): 313–328.

Kramer, M. A. 1991. Nonlinear principal component analysis using autoassociative neural networks. *Aiche Journal*, 37: 233–243.

Laboratory, R. P. I. I. P.; and Meagher, D. 1980. *Octree Encoding: a New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer*.

Li, J.; Chen, B. M.; and Lee, G. H. 2018. SO-Net: Self-Organizing Network for Point Cloud Analysis. In *Proceedings of the 31st IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 9397–9406.

Li, L.; Li, Z.; Zakharchenko, V.; Chen, J.; and Li, H. 2020. Advanced 3D Motion Prediction for Video-Based Dynamic Point Cloud Compression. *IEEE Trans. Image Process*, 29: 289–302.

Mekuria, R.; Blom, K.; and César, P. 2017. Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video. *IEEE Trans. Circuits Syst. Video Technol.*, 27(4): 828–842.

Mekuria, R.; Blom, K.; and Cesar, P. 2017. Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(4): 828–842.

Mekuria, R.; and César, P. 2016. MP3DG-PCC, Open Source Software Framework for Implementation and Evaluation of Point Cloud Compression. In *Proceedings of the 24th ACM Conference on Multimedia Conference, MM 2016, Amsterdam, The Netherlands, October 15-19, 2016*, 1222–1226.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proceedings of the 31st Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, NeurIPS 2017, Long Beach, CA, USA, December 4-9, 2017*, 5099–5108.

Quach, M.; Valenzise, G.; and Dufaux, F. 2019. Learning Convolutional Transforms for Lossy Point Cloud Geometry Compression. In *Proceedings of the 26th IEEE International Conference on Image Processing, ICIP 2019, Taipei, Taiwan, September 22-25, 2019*, 4320–4324.

Quach, M.; Valenzise, G.; and Dufaux, F. 2020. Improved Deep Point Cloud Geometry Compression. In *Proceedings of the 22nd IEEE International Workshop on Multimedia Signal Processing, MMSP 2020, Tampere, Finland, September 21-24, 2020*, 1–6.

Que, Z.; Lu, G.; and Xu, D. 2021. VoxelContext-Net: An Octree Based Framework for Point Cloud Compression. In *Proceedings of the 34th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, 6042–6051.

Schwarz, S.; Preda, M.; Baroncini, V.; Budagavi, M.; César, P.; Chou, P. A.; Cohen, R. A.; Krivokuca, M.; Lasserre, S.; Li, Z.; Llach, J.; Mammou, K.; Mekuria, R.; Nakagami, O.; Siahaan, E.; Tabatabai, A. J.; Tourapis, A. M.; and Zakharchenko, V. 2019. Emerging MPEG Standards for Point Cloud Compression. *IEEE J. Emerg. Sel. Topics Circuits Syst.*, 9(1): 133–148.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Proceedings of the 31st Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, NeurIPS 2017, Long Beach, CA, USA, December 4-9, 2017*, 5998–6008.

Wang, J.; Ding, D.; Li, Z.; and Ma, Z. 2021a. Multiscale Point Cloud Geometry Compression. In *Proceedings of the 31st Data Compression Conference, DCC 2021, Snowbird, UT, USA, March 23-26, 2021*, 73–82.

Wang, J.; Zhu, H.; Liu, H.; and Ma, Z. 2021b. Lossy Point Cloud Geometry Compression via End-to-End Learning. *IEEE Trans. Circuits Syst. Video Technol.*, 31(12): 4909–4923.

Wang, Y.; Wu, S.; Huang, H.; Cohen-Or, D.; and Sorkine-Hornung, O. 2019. Patch-Based Progressive 3D Point Set Upsampling. In *Proceedings of the 32nd IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 5958–5967.

Wiegand, T.; Sullivan, G. J.; Bjøntegaard, G.; and Luthra, A. 2003. Overview of the H.264/AVC video coding standard. *IEEE Trans. Circuits Syst. Video Technol.*, 13(7): 560–576.

Xu, Y.; Lu, Y.; and Wen, Z. 2017. Owlii Dynamic human mesh sequence dataset. *120th MPEG Meeting*, ISO/IEC JTC1/SC29/WG11 m41658.

Zhang, H.; Goodfellow, I. J.; Metaxas, D. N.; and Odena, A. 2019. Self-Attention Generative Adversarial Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, California, USA, June 9-15 , 2019*, 7354–7363.